Three Dimensional Visualisation of Network and Security Log Data

November 5, 2009

COSC460 Final Report by Alexander Chernoglazov
ach136@student.canterbury.ac.nz
Department of Computer Science and Software Engineering
University of Canterbury, Christchurch, New Zealand

Supervisors: Associate Professor R. Mukundan, Associate Professor R. Hunt
mukund@cosc.canterbury.ac.nz, ray.hunt@canterbury.ac.nz

Abstract

This report explores the field of visualisation of network and security log data, specifically the various issues surrounding the processing and graphical display of large quantities of information. A novel network traffic and security alert visualisation technique is proposed along with a method of grouping data in memory based on network addresses and temporal division. An overview of common problems in the area of data visualisation is made along with several suggestions on possible solutions.

Contents

1 Introduction

Visualisation is an important area of research in computer science. It enables human operators of various computer systems to rapidly interpret information that would otherwise be very difficult to analyse. Examples of such information include software metrics, network traffic and various scientific data such as output from a set of electronic sensors. It is known that when done properly, visualisation is a powerful technique that speeds up the process of analysis of large amounts of data [6, 22].

A significant amount of scientific work has been dedicated to visualisation of information with the aim of improving human-computer interaction and providing a more natural means of display of data that is easily interpretable by users. However, the area of visualisation of network and security log data has been neglected to a certain extent, resulting in the low popularity of methods of graphical (in particular three-dimensional) display of information and reliance on scripts and command-line based utilities for processing of large volumes of log data.

For example, a paper by Papadopulous et al [19] has highlighted a reality of the current state of network and security log file processing: a survey of network administrators has revealed that simple open source tools such as *tcpdump* and *snort* are the preferred methods for log analysis in a large number of cases. Graphical analysis was considered either too costly or too limited, not being able to satisfy the requirements of many companies. The same findings have also been reported by Ball et al [1].

This research is motivated by the current lack of inquiry into better methods of visual display of data from log files from network equipment and security software. The demand for high security in computer systems in recent years can be in part satisfied by the increased use of visualisation of network traffic and alerts generated by firewall devices. Early identification of possible threats and anomalies such as port scanning, denial of service attacks and intrusion attempts is becoming increasingly desirable in corporate networks, as such detection enables rapid deployment of effective countermeasures.

However, the issues faced by administrators of a large network are serious and complex. The volume of data generated by modern computer systems is enormous (for a single Class B IP network, a total of one terabyte a day could be generated [1]) and in many cases this information is deposited into a multitude of log files scattered over different folders. Manually searching such a large quantity of data is difficult and tedious and, as a result, log analysis often only occurs after a serious security breach, a network node failure or a related network problem. Log data visualisation software, especially applications displaying information in real time, can be used to diagnose a problem with greater ease and to prevent a potentially serious security situation from becoming catastrophic.

It is therefore very useful for a network administrator to be able to visualise the traffic going through his or her network, and to be visually notified of a suspected intrusion or another suspicious event. There are various kinds of open-source and commercial software aimed at simplifying log data analysis. However, the software available is generally limited to displaying data in two dimensions, by way of plots, charts and graphs.

In order to create better methods of graphically representing log data, it is important to investigate the use of 3D graphs, as they utilise the full potential of computer graphics and human visual perception. Since more than half of the human brain is devoted to the various stages of processing of visual information [9, 17], graphical and particularly 3D methods (to take advantage of depth perception) of representation of data should be researched.

As stated above, this research is focused on the visualisation of network and security log data. In practice this means combining network-oriented output (number of packets and bytes, connections between machines) with data which is of interest from a security point of view (ports and protocols used, automatic alerts generated by firewall equipment).

The two viewpoints overlap to an extent, and in order to build up a complete picture of the current security situation, any security alert analysis must also include underlying network traffic data.

Security alerts are by definition high-level events, and must be caused by the contents of a certain packet or a particular traffic pattern. Therefore, it only makes sense to view these alerts in the context of network traffic by linking the two together. Only then a coherent picture of the state of a network can be built up. This research focuses on exploring this link, attempting to investigate a novel three-dimensional technique of network traffic and security alert visualisation and create a prototype system to illustrate the concepts behind visualisation of such data.

1.1 Report Structure

First of all, an extensive literature survey is conducted in order to fully explain the current state of research in the area of visualisation of network and security log data. Various strategies and techniques, along with their significance and impact are explored.

Next, a prototype log data visualisation system designed during this research is described in detail. Its design is analysed and its performance is tested. The advantages and limitations of the system are described and some potential research directions are discussed.

Finally, a summary of issues in the field of network and security log data visualisation is made along with several suggestions on the possible solutions to them.

2 Previous Work

In this section, previous scientific work is analysed and its importance to the research presented in this report is explained. It should be noted that while there has been a number of publications describing network and security visualisation concepts and systems, the approaches they took were often completely different, and it may be beneficial to establish several common principles that apply to all such systems. These principles are suggested in Section 4.2, while this section is concerned with highlighting the background of this project and the works that have influenced its direction.

2.1 Literature review

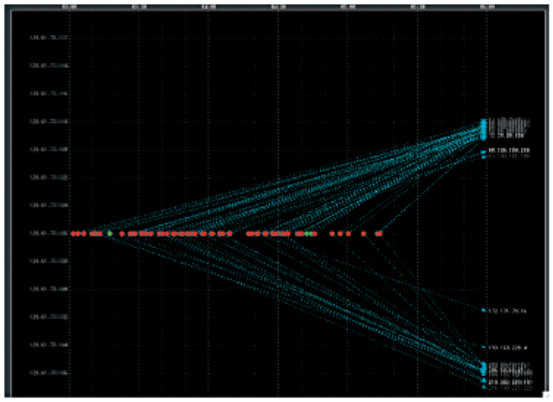
There have been multiple research papers published in the area of network and security log visualisation, but crucially none that bear significant similarity to the proposed three-dimensional graphical representation technique. This section provides an overview of previous work in the area of data visualisation, with particular focus on its applications in the areas of networking and security.

2.1.1 Perspective of Visualisation

Establishing the point of view for the data being visualised is extremely important for any system dealing with network traffic. In practice, it means making a choice regarding what view of the visualised data is most useful for the users and what their principal task is (whether it is finding trends in a large data set, detecting attacks on a network, and so on).

In the area of visualisation of network traffic and security-related events the obvious point of view is that of a network administrator interested in keeping his or her network secure. Therefore, they will be interested primarily in the traffic patterns and occurrences of attacks and suspicious behaviour on their own local network, and a visualisation system needs to account for that by using this local network as a point of reference.

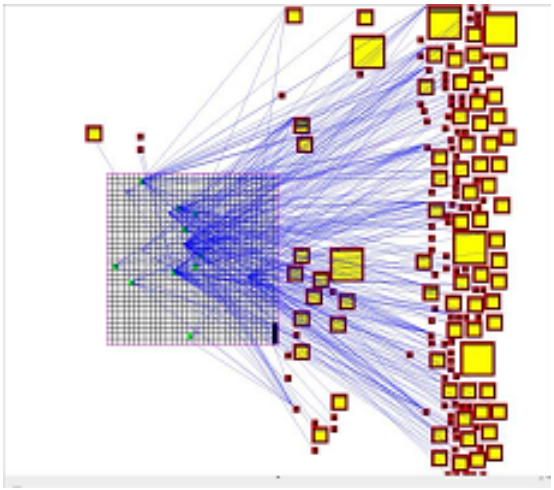
Figure 2.1: RainStorm IDS showing a worm-infected machine attempting to make connections to a large number of remote IP addresses over the course of several hours [6].



All events should be visualised based on their relation to it, as the question of how a certain event impacts the local network is what the system administrator cares most about. This special focus on the local network during visualisation of network traffic and security alerts will be referred to as the *local perspective* in this report. Several successful systems that have been developed using this approach are described below.

In 2006, Conti et al have developed an Intrusion Detection System (IDS) called *RainStorm* [6], which extensively utilised temporal division of data and aggregation of important information. It was used to represent 2.5 Class B IP networks - around 164 thousand addresses in total - on one screen. This was achieved by dividing the 2D graphical display area into a series of rows and columns, with each row of pixels showing alert data for 20 IP addresses. In addition to that, all alerts over a 24-hour period

Figure 2.2: *VISUAL*, a part of the *Network Eye* system developed by Ball et al. 80 hours of network traffic for a network of 1020 hosts are visualised in this window [1]



were displayed as red dots on a black background, which achieved a clear and simple presentation style. By positioning the mouse over an IP address range, the user could swiftly zoom in on a particular range and only view data associated with it, which eliminated the problem of information overload and screen clutter.

RainStorm enabled operators to quickly identify potentially compromised machines and find attack patterns due to showing alerts and suspicious traffic over an entire local network over a 24-hour period. Conti's paper was an important influence on the direction of this research, as it demonstrated the value of aggregation, presented techniques for displaying large quantities of information on a single screen and explained the importance of viewing data from a local perspective.

Ma [16] has conducted work on the subject of visualisation of port scans and probes that typically precede an attack on a network. His approach was quite similar to that employed by Conti et al and allowed the analyst to see connections between hosts on a local network and the external IP addresses that they are communicating with. Depending on the presence of fan-in or fan-out patterns in the visualised data, conclusions could be drawn about the type of probe conducted on the local network.

A paper by Ball, Fink and North [1] reinforces the notion that it is essentially impossible (and impractical) to build up a picture of security information from anything except the local perspective, and every network administrator needs to concentrate on managing the security of his or her own network. This observation is consistent with that of Conti and Ma, and the approach described in all of these papers is the most logical one to take when constructing a network and security log data visualisation system.

Ball, Fink and North's system, *Network Eye*, displayed the local network as a grid and all external IP addresses were marked by squares of various size. Communication between external hosts and local machines was shown by lines connecting the grid squares and the external squares. The size of a square representing an external host determined the amount of traffic exchanged. Users also had the ability to view detailed information for each IP address and find the types of traffic that were used and the ports used to exchange information.

This paper differs from almost all other research published in the same field by the fact that participative design was employed - end users were always involved. A comprehensive user study was also conducted, which highlighted the strengths of the system - the ease with which unusual behaviour was found, the external hosts that have exchanged the most data with local machines and which protocols were most used. Certain types of attacks were also easily identified with the help of *Network Eye* - for example an attacker performing a ping sweep over the entire network caused a clearly visible pattern to emerge in the visualised data.

2.1.2 Immersive Visualisation

While most researchers in the field of network and security data visualisation propose using strictly graphical methods to present information to the user, there has been a paper arguing for greater use of sound in order to create an immersive environment for data analysis.

In 2004, Papadopulous et al [19] proposed a network traffic and security alert visualisation system called *CyberSeer*. The idea behind this system was presenting network traffic flows as two dimensional charts, overlaid on top of a three dimensional graph of all nodes in a network. Next to each node (router or switch) would be a graph of traffic currently going through it (refer to Figure 2.3). This visual representation would also be accompanied by multi-channel spatial sound. In fact, the majority of their paper is devoted to finding the best way to synthesize sound and present network traffic as different waveforms,

Figure 2.3: A proposed visualisation technique for *CyberSeer*, showing 2D traffic charts overlaid onto a 3D map of a network. [19]

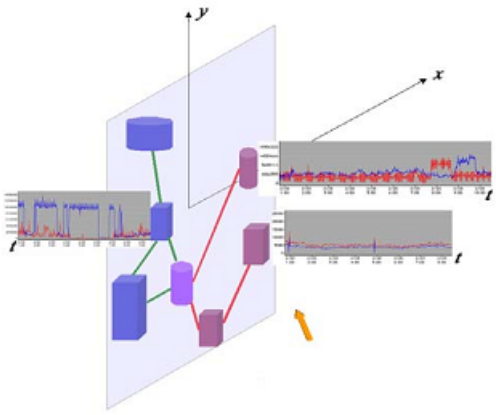
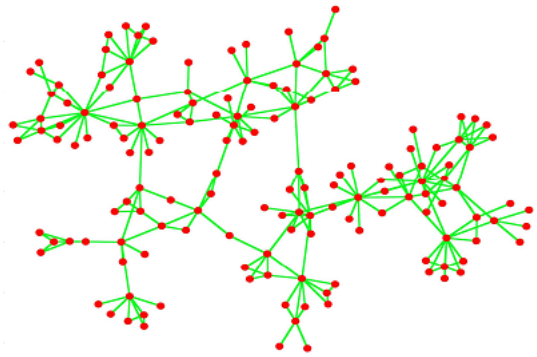


Figure 2.4: Example of a coarsened graph which shows connections between clusters of nodes. 45480 nodes are condensed into 152. By Wong et al [7]



the frequencies and amplitude of which would convey a certain meaning.

The advantages of a completely immersive system that utilises spatial surround sound to relay information about network alerts and traffic conditions are clear. Humans have a good ability to recognise the physical position of a source of sound and orient themselves accordingly [13]. Combined with a large auto-stereoscopic display, this would have created a very effective approach to real-time network traffic analysis. While this immersive system was never created and the authors merely argue the merits of such a solution, the ideas contained in this paper are nonetheless important.

2.1.3 Log Data Processing

As stated above, log files are generally quite large, so appropriate techniques for handling them need to be explored. The issues relating to processing large amounts of data and strategies for visualising them were studied by Shneiderman in his paper *Extreme Visualisation* [23]. Shneiderman mentions that other researchers have created systems for graphical exploration of databases of millions of records [4, 15], and that visualisation of billions of entries will be possible in the near future.

Various methods of aggregation and presentation of such vast quantities of data are discussed by Shneiderman, including merging of related records together, grouping a set of data nodes into so-called *metanodes* (structures showing a more general, high-level view of the data) and use of colour to represent outliers. The author notes that in most cases, a generalised view of data (where some precision and detail is lost) is much more useful to the human operator than a full plot of the same data which does not omit any details. Such representation, in fact, helps by easing the visual load imposed on the user and allowing him or her to identify trends in the displayed data.

Principles similar to the ones above were also suggested by Wong et al [7]. The ideas in this paper, which is not specifically investigating network data analysis, can nonetheless be adapted to the visualisation of network structure and data flows. Wong et al were concerned with reducing the level of detail of data, for example turning a graph of a network containing 46480 nodes into a coarsened graph containing only 152 nodes. The end result was a much clearer visualisation, showing a high-level overview of a network, which showed connections between nodes in a very clear and easily understandable way. A similar approach, which was taken by Perer and Shneiderman [20], involved computing *community structures*, that is, finding a related group of nodes and displaying them as a single metanode on the coarsened graph.

2.1.4 Security Alert Visualisation

A serious problem with most log files is that each entry is separate and physically unrelated. Any two lines, even if they describe different stages of the same event (for example two packets which are both used to conduct an attack on a machine) are not linked in any physical way to previous entries. Therefore,

the ability to parse a log file and reconstruct the logical links between its entries is crucial for a visualisation system.

In practice this means categorising each message according to a certain rule or metric and placing the information it contains into a logical container suited for the purpose of the application. For example, information from an intrusion detection system (IDS) could be categorised by IP address the alerts are coming from, by alert type and so on. Security alert visualisation systems are faced with this issue, as they often have to collect information from a range of network-based sensors.

In 2002, a client-server security information visualisation system called *Tudumi* was created by Tetsuji Takada and Hideki Koike [24], which integrated data from multiple sources such as network traffic, user access and log-in logs. The system summarised available information and graphically presented data to the user in form of several layers of disks - the topmost disk showing the most suspicious login attempts (classified as such by interpreting user-defined rule sets), the bottom disk representing normal activity. The user could filter out uninteresting login attempts and, by concentrating on only several specific connections between usernames and host machines, establish whether an unusual pattern of activity on a network is indeed an intrusion.

The types of alerts that *Tudumi* was designed to visualise were unusual login attempts (a user not logging in from a machine which he or she normally uses) and unexpected connection attempts (for example a computer in another country trying to connect to a host in a local computer laboratory). The authors claim that their system was very successful at prominently displaying such anomalies.

Graphically, the system was somewhat confusing, with some visual clutter being present due to a large amount of login attempts being visualised at once. However, the authors have employed several techniques to draw the user's attention to specific details and visually differentiate between categories of information. As mentioned above, suspicious behaviour was presented as nodes on the top layer, immediately improving its visibility and occluding the less important nodes describing normal behaviour. Secondly, a variety of textures and colours was employed to create distinctive patterns for the events visualised by *Tudumi*. Overall, the system presented a very powerful interface to the user that was difficult to learn, but had the ability to show a variety of security-related data.

A related approach was taken by Hertzog [10] in 2006, who has created a security alert visualisation system which tied individual machines and users to the ports, protocols and applications used. For example, all traffic on port 22 (SSH) was not only looked at from a low level point of view (how many bytes were sent or received), but the types of applications that using this port were also examined.

A graph connecting users, local hosts, applications, ports and remote hosts was constructed, and any anomalous traffic (such as an application not connecting through a port which it normally uses) stood out amongst the connections representing normal and legitimate traffic. This is similar to the idea behind *Tudumi*, as the aim of both systems is to create a visualisation that would give special prominence and visibility to unexpected events and the suppress the display of messages not considered important enough.

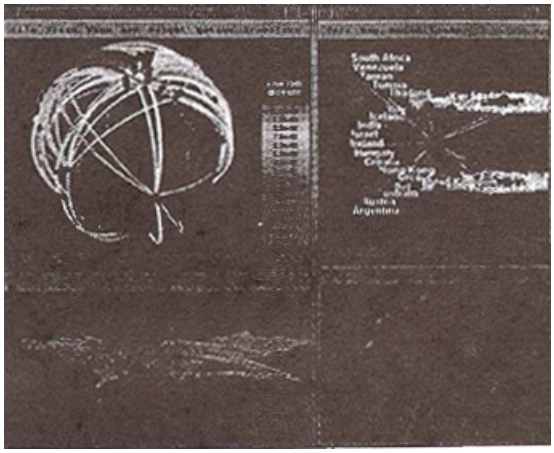
2.1.5 Geographical Visualisation

Geographical visualisations and their relationship to analysis of network data have been explored by several papers which have focused on creating graphs showing data flows between physical locations. The aim of geographic visualisation, in general, is to help the user picture the adjacency of nodes in a network, their spatial location and paths between them [23].

In 2005, Chang et al [2] created a *Deployment Analysis System* (*DAS*) in order to improve the quality of monitoring of a network of field sensors deployed in a real environment (a mountain reserve in California, USA). The primary function of the *DAS* was to help the user determine whether there exists significant congestion in the sensor network, if some routes between sensors are broken, whether a battery in a particular sensor is about to fail, and so on. A topological map was generated by the system to inform the user of spatial locations of sensors, their status and the strength of connections between them.

This work has also described two other important features that are highly relevant to the field of network data processing. One such feature was the display of information in relation to time (called *temporal replay* by the authors), allowing the administrators to pinpoint at exactly which time a certain route was chosen as a means of communication between two sensors, at which time a route became unstable and was removed and so on. The second feature is the modular structure of the *DAS*, with the database being detached from the front-end presentation code and from the modules collecting data

Figure 2.5: *SeeNet3D* by Cox et al [3], showing different views of network traffic flows over the entire world.



from sensors. This greatly improves the extensibility of the system, allowing future addition of more visualisation styles and support for a larger variety of log file formats. Overall, the *DAS* has allowed its users to fine-tune the sensor network, utilise more efficient routing algorithms and even predict trends by viewing the visualised data.

A similar topological approach was taken by Cox et al [3] in a paper describing their attempt at visualising network traffic over the entire world. The questions that their system, *SeeNet3D*, was designed to answer were related to connectivity and traffic flows. The amount of traffic exchanged between countries was of interest and this paper explores several ways of visualising it.

The ideas presented by Cox et al were largely consistent with other research in this area: gathering and categorising log entries was used to only show relevant information and 3D visualisation of traffic flows was employed to get rid of the clutter of two dimensional charts. The system also allowed drilling down, that is, zooming onto a particular region to get a more detailed set of statistics.

Another attempt to visualise network traffic flows by constructing a virtual geographical map was made by Koutsofios et al [12] in 1999. Their system was called *SWIFT-3D* and was designed to graphically display data from different telecommunications network log files and visualise information such as telephone call frequency, quality of service in different areas and inspect the extent of overage provided by various ISP's across the United States of America.

Figure 2.6: An example of temporal granularity by Shabtai et al [22]. The user is zooming in from a low-resolution timeline showing a broad overview of traffic onto a higher-resolution timeline showing a specific time interval of interest.

SWIFT-3D was designed to be flexible and supported multiple log formats. The system parsed a number of log files and aggregated the data into its own internal format. Different visualisation techniques were used, for example static and animated 2D and 3D displays. A very important observation made by this paper is that visualisation is not a linear process: the user should be given the freedom to explore the graphical representation and switch between different views. Overall, the system was successfully used to establish trends in the frequency of telephone calls, determine why there was a large proportion of unsuccessful calls and disconnections in certain cities at particular times of day, and visualise the areas covered by different telecommunications companies.

2.1.6 Temporal Division

One way to group and generalise network traffic statistics and security alert data is by using temporal division. The data that needs to be generalised and aggregated is placed into categories according to the time they've been recorded at. This results in a series of time intervals which are easier to visualise than a large amount of ungrouped data.

Eick, Nelson and Schmidt [5] have been among the first researchers to address the problem of simplifying analysis of large quantities of textual data by presenting it in a graphical format, and temporal division was among the techniques that they employed. In 1994, Eick et al have created *SeeSoft*, a system for visually analysing output data from a switch testing suite. The approach was to split lines in a log file by hour, then only show error messages (thus getting rid of log noise - unimportant messages and information) and use colour coding to sort errors into separate categories. The program turned out to

be very helpful for the users and reduced the time it took to analyse testing logs by 80%.

Another example of temporal division has been demonstrated by Shabtai et al [22] during the creation of their visualisation and analysis tool, *KNAVE-II*. Support for various granularities (such as days, hours and minutes) is proposed, along with the ability to use an absolute timeline, using the calendar and system clock as reference, or a timeline relative to other events contained in a data set.

3 Log Data: Interpretation, Parsing and Visualisation

This section first describes the aim and goals of this research, then its setup, that is, the hardware and software used. Next, a high level overview of the developed application is presented, followed by an in-depth analysis of each of its components and their functionality.

3.1 Research Aim

After highlighting the background of this project, the main research directions and goals of this research are described. The idea behind the application described in this section is that network and security data is connected, and that visualising any security alerts is meaningless if one does not also look at the traffic data behind them.

The idea is to construct a prototype of a network and security log data visualisation system which has the ability to display multiple categories of information from a network device on a set of axes in 3D space. The visualisation performed is static, that is, there is no real-time data processing capability. All graphical analysis will be done from the perspective of the local network, meaning that all traffic and alerts will be visualised in terms of their relation to hosts on the local network, as this is the most logical point of reference [6, 1].

Visualisation is best used to guide the user and bring potentially important messages to his or her attention [16], so the aim of the proposed system is not to predict network activity or suggest what attacks may be incoming in the near future. The goal of this system is to provide a broad overview of condensed information about network traffic and help its operator to recognise patterns by categorising and filtering data.

Such a representation will, ideally, enable the user to determine at a glance which computers on a network are the most active and what types of traffic are most used. Alerts can be presented visually, indicating which machines are coming under attack, at what point in time these attacks are happening and what types of traffic are being used to conduct them.

Another capability of the proposed system includes displaying data over a period of time by way of dynamically animating the graph that is displayed. Since motion is a good way of attracting the user's attention to an area or object [30, 31], animation of data will enable the operator to find out at which times of day the network is most saturated and when certain machines are most active. Animation was also utilised by Ball et al and Chang et al [2, 1] in their visualisation systems, and is considered a viable way of reducing the cognitive load on the user [26].

3.2 Hardware and Software

First of all, the hardware and software setup used during this research will be briefly described.

An Intel Core 2 Quad Q8200 CPU was used, along with 4GB of DDR2 RAM and an NVIDIA GeForce 9500 GT graphics card. Microsoft Windows XP was the underlying operating system. The application created during this research was written in the C++ programming language.

3.2.1 OpenGL and GLUI

OpenGL [8] was chosen as the graphical API used to draw the visualised data. OpenGL was preferred because of its cross-platform compatibility and ease of use. GLUT [25], the OpenGL Utility Toolkit was used to provide auxiliary functions such as those used for window drawing and GUI callbacks (for example, key presses and mouse movement).

One of the disadvantages of combining C++ and OpenGL is the lack of a standard GUI API, which

Figure 3.1: 2 lines from a firewall log data file used during this research

```
5;15Apr2009;23:46:32;132.181.3.233;account;accept;;eth0;inbound;;;
    Default_Protection;15Apr2009 23:45:52;15Apr2009
    23:45:52;0:00:00;2;88;1;1;1;1;44;44;44;44;eth0;eth0;eth1;eth1
    ;132.181.0.5;124.171.216.203;udp
    ;1513;1513;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
6;15Apr2009;23:46:32;132.181.3.233;account;accept;;eth0;inbound;;;
    Default_Protection;15Apr2009 23:45:44;15Apr2009
    23:45:44;0:00:08;4;176;2;2;2;2;88;88;88;88;eth0;eth0;eth1;eth1
    ;132.181.0.1;60.242.172.75;udp
    ;1084;1513;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Figure 3.2: Part of the graphical user interface created with GLUI for the visualisation system described in this report. Buttons, text boxes, check boxes and separators are visible. Refer to Figure A.1 in the Appendix for a full view of the program's GUI.

would allow a program to have both an OpenGL viewport where a scene might be rendered and a standard user interface which could be be used to control the display of the scene and issue commands. Due to this lack of standardisation, custom libraries must be used. GLUI [14], the OpenGL User Interface Library, was the library selected to provide a standard GUI to the user, with widgets such as buttons, text boxes and check boxes being present. Figure 3.2 illustrates some of GLUI's capabilities.

In the application, GLUI managed both the visualisation window and the GUI presented to the user. This library was used to unify these two parts and present a single coherent interface to the user.

3.2.2 Log Files

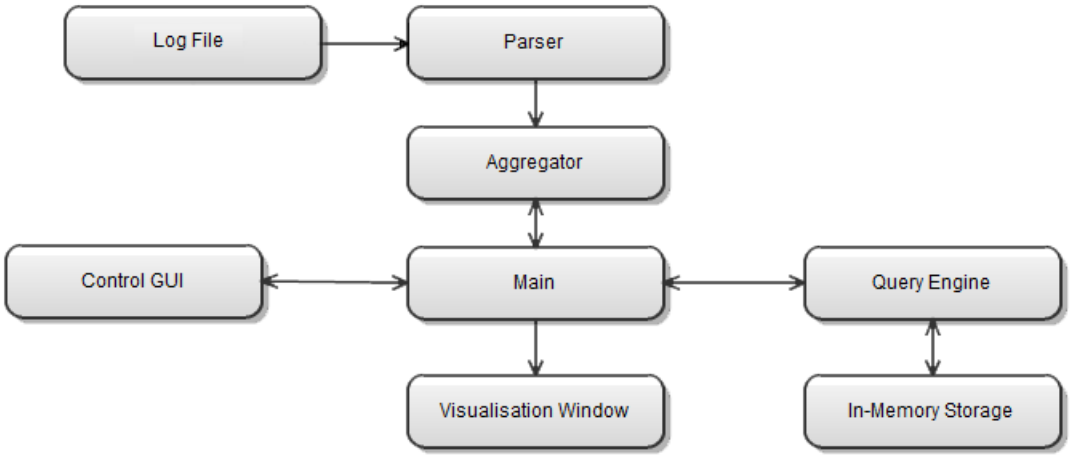
The log data files used during this research were obtained from a CheckPoint firewall operating on a Canterbury University computer network. These files used contained all information that is expected to be present in a firewall log file - entries such as IP addresses of both communicating parties, amount of bytes and packets exchanged, names of incoming and outgoing interfaces, protocol types and so on.

The only information that was removed was the sensitive user data, such as internal Canterbury University network user codes, each of which are unique and can be tied a person, thus compromising privacy. The above restriction did not impact this research in any way, as the main focus was very general: developing a set of techniques for presenting network traffic and security alert data in a three-dimensional format.

The log files analysed were recorded over a 24-hour period and contained highly detailed information about the operation of a part of the Canterbury University computer network. Therefore they were deemed to be a highly realistic representation of a real-world data set.

Not all data contained in the log files examined was processed. Among the most relevant and useful fields were: the source and destination IP addresses, the protocol types (both low level such as TCP and UDP and higher level such as HTTP and DNS), the time the communication took place and the alert(s) generated.

Figure 3.3: High Level Architecture and Control Flow



3.3 High Level Architecture

The proposed visualisation system consists of a number of interconnected modules that each perform a certain task. This section describes the general structure of the application and its overall functionality.

Figure 3.3 presents a general overview of the structure of this application and its control flow. All controls are issued by the user through the control GUI panel. This panel is integrated with the main visualisation screen. Any command sent through the control GUI is processed by the main module of the application. Here a decision process takes place, classifying the request and taking appropriate action.

The first type of request is that for loading a log file. Once such a command is received, the parser is called, which operates on a user-specified log data file. During parsing, the aggregating function is called which classifies and separates log data according to certain parameters which will be discussed later. The aggregating engine sorts the data received from the parser and places it into a series of specialised structures held in RAM, as explained in Section 3.4.3.

Thus, a large in-memory database of all content deemed relevant for the purpose of this application is built up. The idea behind this large set of dynamically allocated container objects is to make query execution significantly faster than would otherwise be possible with the aid of a hard-drive based database.

After the loading and aggregation processes are complete, the user can issue commands through the control GUI to query the in-memory database for assorted information. The querying engine performs a series of lookups in the storage structures and extracts relevant information. For the purposes of this prototype application, which is intended as a proof of concept, the types of queries that can be done are restricted to a set of pre-defined requests, which are described in detail in Section 3.4.3. After the appropriate look-ups are performed, the completed graph is drawn in the visualisation window.

The above is, in general terms, a broad overview of the system's functionality. It has been designed to be as fast and efficient as possible, handle large amounts of input data (possibly totaling up to 2 GB) and produce accurate and complete graphs according to user queries.

3.4 Functionality

This section describes, in detail, the functionality of the various components of the proposed system. Their purpose and internal structure is analysed and the concepts related to visualisation of network and security data that are illustrated by these components are explained.

3.4.1 Parsing

The parsing module of the proposed system was designed to be modular, in theory allowing anyone with access to source code to write a small class which

can then be instantiated and used in place of the default one. Ideally, multiple parsers will be integrated into the application to seamlessly provide required information upon request from a user, although this functionality has not been implemented in the current iteration.

This approach was chosen with the purpose of solving one of the most serious problems facing any log visualisation application - the lack of compatibility between different log formats, as mentioned in Section 2.1.3. Several papers [24, 12, 2] emphasise the the need to support multiple log file formats and to keep a custom internal database which combines data from all sources.

Currently the parser reads the specified file line-by-line and extracts information from it, breaking it up into separate fields and sending the resulting data back to the aggregator. Once a data structure containing the extracted data from a single log message is received, it is sorted according to the algorithm described below.

3.4.2 Aggregation and Classification

The main grouping and sorting process takes place at the same time as parsing, as the two functions are logically linked - in order to make the program faster and less resource intensive, simultaneous reading of the log file and grouping has been implemented.

One pass of the aggregation algorithm is done every time the function receives a data structure containing all important information extracted from one entry of a log file. The data structure is sent by the parser and contains the exact time of the communication, the source and destination IP addresses, the alert generated (if any), the amount of bytes and packets that passed between machines and the general (layer 4) and specific (layers 5 and above) protocol types. This decision is based partly on the work by Hertzog [10], which identified a set of *<timestamp, user, source host, application, destination host, destination port>* to be the most pertinent information for a security data visualisation system.

3.4.3 In-memory Data Structures, Querying and Data Retrieval

The aggregating module groups information extracted from log files according to certain principles which have emerged as being the most logical ways of classifying information during the research.

Figure 3.4 shows the structure of the in-memory data container used by the application. On a high level, it can be described as a series of 4 hashtables, traversal of which until the end node results in the traversing function obtaining full information about the communication between a local and a remote address at a certain period in time, with the protocols used in communication and other auxiliary information (such as alerts) also becoming known.

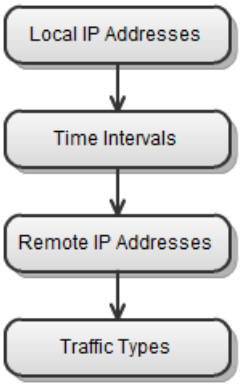
Internally, there exists one large hashtable of all local IP addresses, as viewing security-related data is best done from a local perspective [6, 1]. A search for an IP address in this table results in a data structure containing the local IP address and a hashtable of all the time intervals when this machine has been active being returned.

Variable-precision grouping is used in this application, with the time interval being a flexible variable, which can be changed according to the goals of the visualisation. The idea of a time interval serves as a simplifying measure, which aggregates all traffic within a certain time period (for example, 6:00pm to 6:15pm, 6:15pm to 6:30pm, and so on) into one group in memory, decreasing storage requirements. A similar approach has been used by Ma [16] and Shabtai et al [22] for alert visualisation.

Naturally, when log messages that are precise to within milliseconds are placed into much less specific containers, some data is lost, but that is necessary for the balance between storage space and precision to be achieved. Larger time intervals being used will reduce precision (that is, it will become more difficult to establish at precisely which point in time communication between two IP addresses occurred) while decreasing storage requirements, while smaller intervals will cause the reverse to happen. Precision will improve as the length of the interval is shortened, but at a cost of rapidly increasing memory usage.

The hashtable of time intervals retrieved can then be queried to find whether it contains an entry for an IP address of interest. In case of a positive result, another data structure is returned, containing the remote IP address, a Boolean variable set to true if any alerts have been alerts generated for that IP address, and a hashtable of all the different traffic types used during communication with the local IP. If this newly obtained hashtable of traffic types is traversed, exact amounts of bytes exchanged using different protocols can be looked up and the number of packets sent and received can be easily found out. Alerts that may be generated by the firewall are stored together with the amount and type of traffic, so it is always possible to identify which traffic type was used to attack a host on a local network.

Figure 3.4: Structure of the hashtable-based in-memory storage system. Local IP addresses are contained in the first hashtable, with each address linking to a table of time intervals. Each entry in a table of time intervals links to a hashtable of remote IP addresses which have communicated with this local address during this time interval. Each remote IP address links to a table of all traffic types used, which contain detailed statistics about the packets and bytes exchanged during communication along with any alerts generated during it.



An example of the query process is provided below. Suppose a local IP address 192.168.2.1 communicates with an external IP address 202.5.67.13 at 6:13pm. Their communication is made over the TCP protocol; it uses 100 incoming packets, 100 outgoing packets and 100,000 bytes are transferred each way. This exchange generates one alert. Therefore, in the table of local IP addresses, 192.168.2.3 will be looked up, and this communication will be recorded as being during the time interval between 6:00pm and 6:15 pm. Next, 202.5.67.13 will be inserted into the hashtable of all remote IP addresses that have had contact with the local one during this time interval. The TCP traffic type will be inserted into the list of all traffic types used by 202.5.67.13, and the incoming and outgoing packets and bytes will be recorded, along with the single alert.

While the query process may appear complex, in practice any IP address can be found with relative ease, the speed of this process has been found to be adequate, and, most importantly, traversal does not consume any additional memory resources as no significantly large data structures are created in memory during querying and retrieval operations. Its performance is analysed in Section 3.6.

The querying process should be specialised for the purposes of the application being developed. In this case, it has been decided that high performance and low memory usage is more important than minute detail, and therefore the interval has been fixed at 15 minutes, striking a reasonable balance between precision and storage requirements.

Querying itself is controlled through a GUI, which is used to issue commands. There are three types of queries currently supported by the system. The first type displays all traffic and alert contained in the database (that is, the sum of traffic over all time intervals), the second retrieves all information and divides it by time interval and the third is a query for all traffic associated with a single local IP address. Translated into tasks which may be carried out by a network administrator, these functions might read, respectively: ”show all log data for the local network”, ”show log data for the local network between time X and time Y”, and ”show all log data for local IP Address Z”.

3.4.4 Visualisaton

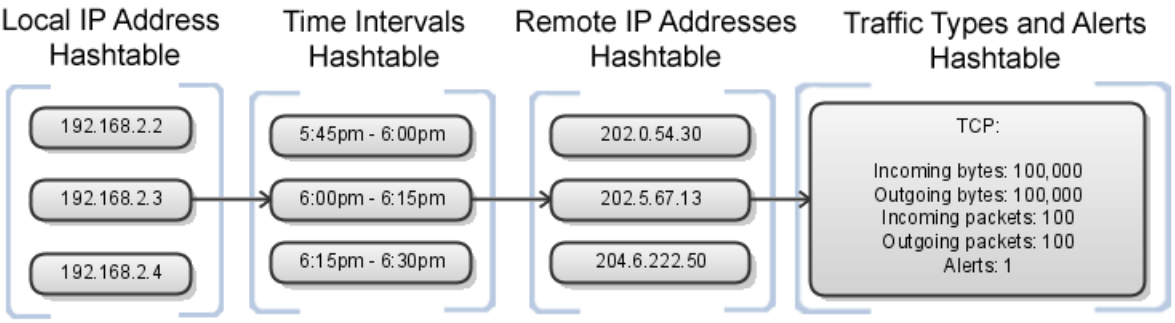
In this section the most significant part of the proposed application is described. As explained above, the visualisation performed is three-dimensional and does not use realtime data. Figures A.2-A.5 in Appendix A provide high-resolution screenshots of the application and the types of visualisation it can perform.

General Structure

The basic structure of the 3D graph proposed in this report is as follows. A number of vertical bars originating at the floor of the visualisation area is drawn. The height of a bar represents the amount of network traffic, and its spatial location indicates the type of traffic and which IP address it originates from or is addressed to. Refer to Figure 3.8 and to screenshots in Appendix A for a full explanation of the proposed technique.

In order to create a point of reference for the user, a neutral-coloured grid has been placed on both the bottom and the back of the visualisation area. This grid clearly marks the floor which serves as a base from which all bars representing network traffic originate. The back of the visualisation area not only serves to give a point of reference to the user, just like the floor, but is also used to display information. The back is used to display the total amount of traffic sent or received over all traffic types for one

Figure 3.5: Insertion of data about IP address 202.5.67.13 which communicated with local IP address 192.168.2.3 at 6.13pm using TCP traffic and generating one alert.



IP address, as shown in Figures 3.9 and A.2. This way, the user is made aware of both the total amount of traffic used by a host, and of the more precise usage data which is broken up by traffic type.

Navigation

An important aspect of any program which creates a 3D scene is navigation. In general, navigation in a 3D virtual world can be difficult and confusing for the user who can get disoriented and lost while browsing it [3]. A variety of control methods is available, for example a free-form movable camera, such as those used in aircraft simulators and first person shooter video games. However, a scheme like this is inappropriate in this case, as the aim of the proposed application is to be accessible to all personnel who come in contact with the management of a computer network. No previous training or experience is assumed and thus the system's navigation and interaction paradigms have to be designed accordingly.

To create a simple yet effective navigation method, a system where the camera moves along a fixed path has been implemented. The camera is fully rotatable using the mouse, which provides the user with the ability to view the visualised data from multiple viewpoints. The user moves the camera with the help of arrow keys on the keyboard, with the presses of specific keys being translated into corresponding camera movements.

The camera can be moved both along the x and the z-axis, to gain a better view of rows located further away. However, as there is always a possibility of disorientation in 3D space, to ease navigation and to tie the camera to the ideal path, the notion of teth-

Figure 3.6: Proposed 3D visualisation technique. IP addresses are located on the x-axis, with different traffic types being placed along the z-axis. Bars are drawn inside the squares on the floor grid, with bar heights indicating the amount of packets or bytes sent and received. The XZ plane is used to draw the floor grid, the XY plane is used to draw the back wall grid.

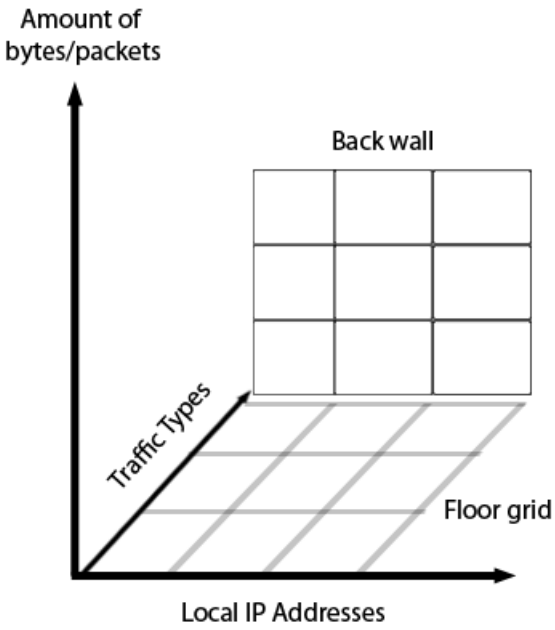
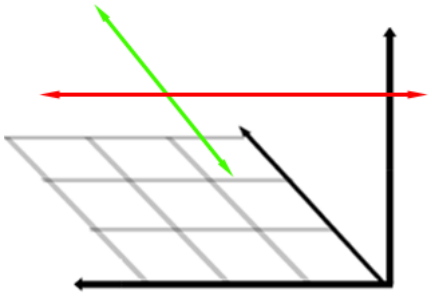


Figure 3.7: The camera mostly moves along the x-axis, with the camera path along that axis being marked in red. This is the pre-defined path which is best suitable for viewing data. Camera movement along the z-axis is also possible, and is marked by a green arrow. Should the camera move off the pre-defined path, it will be pulled back after a timeout of 20 seconds.



ering is introduced. It involves dynamically pulling the camera back onto the predefined path once the user has stopped moving it, after a timeout of 20 seconds. This is done with the purpose of preventing the user from getting lost and being unable to return to the path that is best suited for viewing the visualised data. The user is able to toggle automatic tethering on and off. Figure 3.7 explains the movement of the camera.

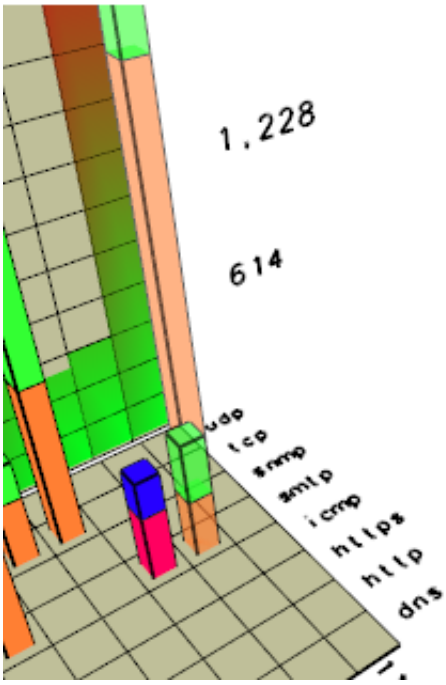
Scaling

It is expected that traffic amounts will vary greatly over the entire range of IP addresses being examined. It is not unusual to have a spike in the visualised data which is larger than the surrounding data by an order of magnitude. If such data is displayed on the graph with no scaling, peaks will generally not be visible. Therefore, in order to avoid such visual discrepancies, scaling has been introduced, where all bars' heights are normalised using division by the height of the tallest bar. Using this scaling technique it is always possible to see the entire height of a bar from any point on the standard camera path.

Alert Visualisation and Colour Coding

Alert visualisation is done in the following manner: bars representing time intervals during which alerts have been received are indicated by a rapidly mor-

Figure 3.8: A bar representing an alert caused by SMTP traffic, surrounded by normal traffic. Normal traffic is also colour coded: green for outgoing and orange for incoming.



phing colour pattern that is designed to contrast with the constant shading of the bars representing normal traffic.

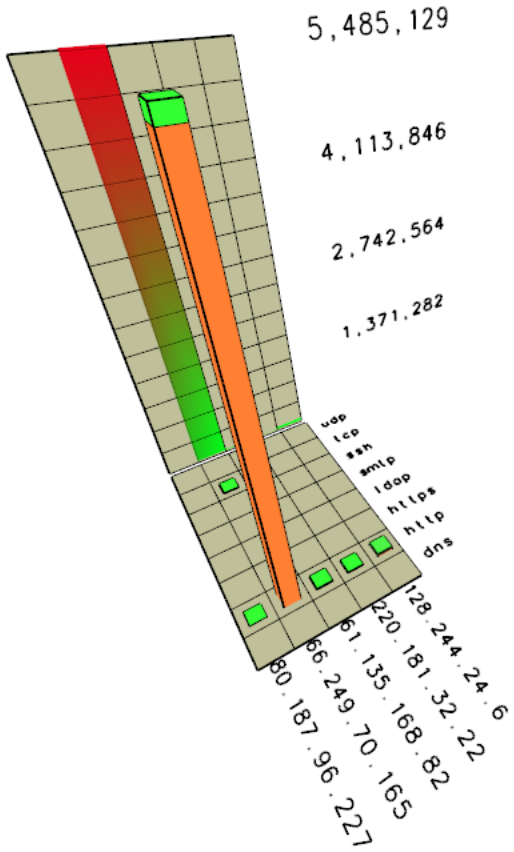
The colour changes are specifically designed so that a bar representing an alert appears to be pulsating. Rapid changes in colour or rapid emission of pulses of light is already commonly used by emergency services (for example, rotating emergency lights on Police vehicles) and in industrial settings to warn of danger. Therefore it is expected that users will be able to apply their previous knowledge to a new domain and recognise a well known signal of danger.

Color coding has been used to differentiate not only between traffic causing an alert and normal activity, but also between outgoing and incoming traffic. As shown in Figure 3.8, green is used for outgoing traffic and orange for incoming traffic.

Filtering

As mentioned before, filtering is an essential component in any visualisation application that is greatly beneficial for reducing visual clutter and giving the user control over the presented data. This system incorporates a simple filtering module that allows the

Figure 3.9: An example of filtering and zooming in on a specific IP address. Here all communication for one local IP address is shown, with several traffic types (such as FTP, ICMP and SNMP) being filtered out. It is clear that the most traffic was exchanged with 66.249.70.165, using the HTTP protocol. Around 5 megabytes was received and around 500 kilobytes was sent.



user to remove any type of traffic from the graph, as shown in Figure 3.9.

It is also possible to view more precise and detailed data by highlighting an IP address and performing a specific IP address lookup. Detailed statistics are then brought up in graphical form. These statistics show the amount of traffic that has been exchanged between the chosen local IP address and all remote hosts it had contact with.

In addition to filtering by traffic type, the user can choose to view alerts only (regardless of the type of query performed), eliminating all IP addresses for which no alerts have been generated. Only bars which belong to an IP address which has an alert associated with it will be drawn on the screen. This greatly reduces the number of objects displayed on the screen at one time and immediately draws the user's attention to those local addresses that have come under attack.

Animation

As noted above, animation is also utilised by this visualisation system. When a user performs a query for full traffic and alert data separated by time intervals, the bars representing traffic during each interval are drawn dynamically and animated over the course of several seconds. The user is able to adjust the animation speed. Figure A.4 shows one such query in progress.

3.5 Testing and Evaluation

In order to assess the performance of the system proposed in this paper, a sample data set was loaded. The file contained information about around 5 hours of operation of a local network of 518 IP addresses. The time it took for the log data file to be loaded and processed was measured, along with the average query times and average memory usage. Results are described in Section 3.6.

A short informal evaluation has also been conducted at the end of this research, attempting to discover the advantages and shortcomings of the proposed three-dimensional visualisation method. The conclusion was that while the application lacks some desirable capabilities such as the ability to process real-time data and does not have an option to select sections of a graph for close-up viewing (it is only possible to view data for an entire network or for an individual IP addresses, but not for groups of IP addresses), it is a sufficiently advanced system that acts as a proof of concept, showing the viability of the proposed 3D technique for network traffic and security alert visualisation.

3.6 Results

The application's performance and its average memory usage are analysed in this section.

3.6.1 Performance

The performance of the proposed system has been evaluated (refer to Section 3.2 for test machine specifications) and the following results have been obtained.

First of all, in order to perform parsing and aggregation on a 220 MB log data file, the system took,

on average, 303 seconds, or approximately 5 minutes. Secondly, a query for data separated by time intervals took, on average, 1.77 seconds. Queries for full data without separation by time intervals were somewhat faster, with the average being 1.52 seconds.

On the test machine, the system ran at 60 frames per second most of the time, except during animations, during which a large number of objects was constructed. This lead to a slow down in the frame rate, which dropped to an average rate of 15 frames per second.

3.6.2 Memory Usage

During testing, the following memory usage has been observed. When a 220 MB log data file was loaded, the average usage was close to 15 MB before loading and 225 MB after loading. Therefore, around 210 MB of RAM was used to store information about the traffic patterns and alerts over approximately 5 hours in a network of 518 machines. However, along with the primary in-memory storage containers, some space was taken up by the various supporting structures, such as the containers used to hold information about the shape and appearance of objects drawn on the screen.

It has also been observed that on average, for this particular data set, each machine on a local network had contact with 32.7 different IP addresses over a 15 minute period. Each machine was active, on average, for 9.27 15 minute intervals out a total of 23. On average, 1.1 different traffic types were used to communicate with each IP address. Therefore, in this network

$$518 \cdot 9.27 \cdot 32.7 \cdot 1.1 = 172723 \qquad (3.1)$$

data structures containing traffic and alert information will be used.

In conclusion, at a certain point (depending on the host computer's resources) there will be a severe strain on the resources of the machine this system is running on. Currently, it is expected that log files in excess of 1 GB in size can be analysed without significant problems. However, in the future it is hoped that increasing amounts and decreasing cost of RAM will be able to counteract this problem.

4 Discussion

This section discusses the results obtained over the course of this research and explains some of the limitations of the proposed visualisation system, making note of possible enhancements and future research directions. A summary of visualisation principles and suggestions on issues to consider during the creation of any network and security log data visualisation system is made. Finally, an overall conclusion is presented which briefly summarises the findings and contributions of this project.

4.1 Limitations

First of all, the obvious problem of the absence of a comprehensive user study is addressed. It was not feasible to perform a large-scale analysis involving a significant number of users during this research, and therefore the evaluation was necessarily limited. However, the system developed was a prototype and not intended for any real-world use. Several extensions and enhancements to it are proposed in Section 4.3.2.

This problem is in fact common in the area of data visualisation where a wide variety of systems have been designed, but very few user studies have been performed. For example, a survey of program and algorithm visualisations [28] has found that only half of the systems described had any usability evaluation at all. Out of several papers on security alert visualisation reviewed during the course of this research, only one (by Ball et al [1]) had a formal user study, which consisted of a group of only 8 participants.

The performance of the proposed visualisation system tended to drop dramatically during animated display of temporally divided data, which sometimes made viewing and navigation difficult. Long loading times were also unacceptable. Finally, another limitation of the proposed system is its inability to process arbitrary, user-defined queries. This can be solved by removing the proposed custom in-memory storage structures and substituting them with a hard-drive based database. Connecting this application to an SQL-based database will likely reduce performance, but gain flexibility and circumvent the memory limitations that are currently present in this software. This is a very good illustration of the trade-off between speed and flexibility which must be dealt with by any visualisation system.

4.2 Visualisation principles

A combination of findings from the literature review and the research described in this report has been used to determine the correct approach to take when creating any network and security log data visualisation system. Several principles and issues that should be considered during visualisation of such data are suggested in this section.

These principles were gathered and summarised based on previous research, setting out a basic set of guidelines that it is hoped may be useful for a designer of a network and security log data visualisation system.

4.2.1 Log Data Processing and Aggregation

First of all, the issue that has received a large amount of attention by previous researchers will be looked at. It is the problem of processing immense quantities of textual data that is generated by various devices. As the total amount of log files that may need to be analysed in a large network is generally expected to be of the order of gigabytes, a certain amount of compression and loss of precision is tolerable, and in fact necessary in order to generate any meaningful visualisation of this data.

The exact nature of this loss is dependent on the data being processed and on the purpose of the visualisation. A system providing a high-level, general overview of a data set can generally afford to make a number of simplifications and omissions with no discernible negative impact. A good example of this principle is presented in a paper by Eick et al [5], which was mentioned in Section 2.1.6. All data was

split into categories based on the hour of day. This was necessary, and yet this obvious loss of precision lead to no adverse effects, and, on the contrary, increased the efficiency of analysing testing results by 80%.

The need for simplification of complicated log data has also been observed by other researchers [19, 24, 23], as was the conclusion that sacrificing precision for the sake of clarity of visualisation was highly beneficial in many circumstances. Trends were presented much more clearly and connections between nodes or groups of nodes in a network were more obvious than in a detailed graph which showed a large amount of connections.

It must be noted that there will be situations where access to the most accurate, verbose data is paramount and where aggregation leading to loss of precision is not tolerable. Certain network alerts such as exact times when a machine came under attack may need to be known with great accuracy in order to correlate them with intrusions and suspicious events that happened elsewhere the same time.

The following conclusion must be drawn: any visualisation system should be developed while keeping in mind the general principles, but at all times concentrating on the problem being solved (that is, the data being processed and the purpose of its visualisation) and its unique constraints and requirements.

4.2.2 Precision Level and Filtering

Being able to change the level of precision of the displayed data and dynamically include or exclude certain types of data is important, as it allows the user to interactively focus on only the important information, and assist him or her in making correct decisions based on the visualisation. Changes in level of precision may also help identify trends, as some may be present at a high level (for example a tendency for the general level of activity in a network to decrease in early morning and increase during the day) and some at a much lower level (for example a specific machine coming under attack at 10 minute intervals).

As described above, any visualisation system must maintain a certain balance between showing as much relevant data as possible to the user and generalising and simplifying data. It is simply not possible to show all the information that is contained in log files due to the extremely large quantity of content, most of which is highly likely to be unimportant to the user.

For example, if an output file from a firewall device is being processed with the purpose of visualising security alerts, certain details in it that are present in every log message may be of lower importance, such as the interface on which traffic comes in. While this information may be required if data flows in a network or connections between networks are being analysed, it is of lesser significance from a security point of view, and therefore may be omitted from a graph. Details like the IP addresses of external hosts that communicated with local machines and ports and protocols used are of much greater interest and should be focused on.

As visualisation can be thought of as an iterative process [16, 27, 11], one where each successive iteration adds precision (in other words, zooms in on a region), being able to change the level of detail is absolutely vital in any visualisation system. A good example of this principle is the *Rainstorm* IDS system [6]. By default, the information was aggregated into groups of 8 IP addresses each. This generalisation (which allowed around 164 000 IP addresses to be shown on a single screen) was combined with the ability to zoom in on areas of interest at any time and view detailed information about security alerts for each IP address.

This fusion of both high and low level views produced a very effective system that reduced the visual load imposed on the user and yet allowed a high degree of control over the information displayed. Cox et al [3] have employed the same technique, where a high level 3D overview of traffic over the entire world was presented to the user, along with the option to drill down and zoom in on lower level nodes.

Filtering is also an obvious requirement to any information visualisation system. The ability of the user to choose what should be shown and what should be omitted is vital for the proper functioning of any application that handles a large amount of data and graphically presents it to the user. Filtering can take several forms, such as the removal of unnecessary information while parsing log data, as described above, excluding a certain category of data from the final graph (for example, specific protocol types or port numbers) or visually occluding part of the screen containing irrelevant messages, relegating it to a poorly visible area of the screen. In conclusion, the type of filtering that should be implemented in a system is entirely dependent on its intended uses.

4.2.3 Visual Presentation

The issues behind visual presentation of data which are relevant to fields of computer networking and security will be examined here. As was mentioned in previous sections, on a very basic level, it is important to decide what information will be extracted from log files and presented, what sort of control will be given to the user over its presentation and how the data will be aggregated. The actual graphical side of the visualisation needs to be given a lot of consideration as well.

No single approach to visualisation of any kind of data can be considered "correct", and all such systems must be examined in the context of the data being processed and the needs of their users. However, there are some common ideas that appear in most publications regarding network and security log visualisation. The most successful visualisation engines were those that managed to present information in a clear and concise way, without overloading the user with unnecessary and unrelated messages or alerts.

Many diverse styles of graphical presentation, from 3D geographic network displays [3] to grid-based 2D alert visualisation [1] to a 2D graph with Chernoff Face glyphs for visualising DNS traffic [21] have been attempted. All have their own merits and disadvantages, such as the tendency of 2D charts to get cluttered as more and more information is added to them or the occlusion and navigation issues facing 3D graphs, where larger bars can cover up smaller ones and prevent important features from being noticed and where camera movement is not always natural.

The type of visualisation, whether it was constructing a topological map, presenting a set of 2D charts or an interactive 3D display to the user is not what determines whether a visualisation system is useful or not. Instead, it the use of the chosen style that is important because the decisions regarding omission and generalisation of information are crucial and must be done in any system, regardless of the graphical style employed. Cognitive economy (minimising the load on the human cognitive system) is an important principle, with less information presented to the user often being more effective than a large amount which tends to overload the user [26].

A coherent visual style (that is, the application behaving in an expected manner, no matter what the query is) and ease of navigation are more important than the complexity of graphical presentation. Providing multiple views of the visualised data is important along with dynamic feedback which is clearly given to the user [18].

4.2.4 Compatibility and Extensibility

Among the issues to be considered during design of a network and security log data visualisation application is the important question of compatibility. While there is significant number of commercially available graphical analysis tools for firewalls and other network and security equipment, they often possess limited capabilities, only being able to process a certain log file format that is generated by the equipment produced by the same company.

In the real world, however, log data can come from various devices that may each have a separate and distinct way in which they format files. Thus it is greatly advantageous for an application to be able to open and read files of different formats, using a specialised parser for each one and gathering the extracted data into a single internal collection, the format of which can be tailored to suit the needs of the application. Depending on the application, performance may be important, or level of detail of data, and so on. Several successful systems, such as *SWIFT-3D* [12] and *Tudumi* [24] have taken this approach.

Extensibility and modularity are the solutions to the problem of compatibility. A single monolithic program is not flexible enough to respond to changing conditions, such as emergence of new technologies or installation of new equipment on the network (which may generate log files in a new, unknown format). A good example of a visualisation application that has been designed from the start to be modular is the *DAS* by Chang et al [2]. It relies on an extension system for its operation, namely on the input extensions for providing data from log files and on the output extensions for providing different means of graphical display. As requirements change, more input extensions can be created to supply the system with log data and more output extensions can be written to improve its visualisation capabilities.

In conclusion it is suggested that modularity is the best approach to the design any visualisation system. Instead of attempting to predict which log formats might be used in the future, it is much better to let the users plug their own parsing modules into the visualisation application.

4.2.5 Performance

Another area of interest to the field of network and security log file analysis is the performance of a visualisation system. With the huge volumes of data which could be processed daily, performance becomes a serious concern and ways of ensuring smooth operation and rapid response times must be found. There is no simple answer to that, but generalisation of gathered data is the only real solution. Combining a large amount of highly detailed messages into one more general meta-message not only improves performance, but also enhances the ability of human operators to recognise trends in data, as shown by Shneiderman [23].

Some ways of increasing performance were suggested by Koutsofios et al [12] who have designed a system capable of processing several days' worth of information, with each day yielding around 15 GB of data. Streaming data from the hard disk by multiple concurrent readers was utilised to improve performance and on-the-fly compilation of user-defined query expressions into C code was used to speed up the process of filtering data from log files. This way, due to queries being dynamically generated and data being gathered from hard disk-based storage, no unneeded information was kept in memory, and only that which was relevant to the user's request was retrieved.

As is the case with most other systems, the above approach is not necessarily correct. In certain cases, such as for processing of smaller amounts of data, it may be possible and indeed desirable to keep as much as possible in memory for rapid retrieval.

4.2.6 Summary of Principles

There have been many attempts at network and security log data visualisation, but there exists a number of common elements in most of them. Below a summary of the most important points is presented.

- Splitting data into groups of related messages with the purpose of reduction of overall complexity is vital. This generalisation allows easier identification of trends by human operators and leads to reduction of visual clutter. Grouping can be performed according to a variety of rules, for example by local or remote host address, traffic or alert type or time interval.

- Modular design leads to extensibility, which is a highly desirable attribute for any visualisation system.

- Providing a way of dynamically interacting with the system is crucial, as letting users explore the visualisation is the best way to ensure that trends will be noticed. Zooming in and changing the level of detail is one such technique and is a very important component of any visualisation system.

- The graphical style (2D, 3D, 3D with 2D overlays and so on) is on the whole less important than specific decisions made about its use. Any style can be used effectively if proper human-computer interaction and design principles are followed. Animation can be used effectively, but it is not essential.

- A network and security log data visualisation system needs to be designed for a given task, focusing on the particular problem that needs to be solved and adapting known methods to solve it. There is no single technique in the field of data visualisation that is applicable in all circumstances, and attempts to make a system as generic as possible are likely to only reduce the effectiveness of such a system.

All of these principles have the common aim of helping the user of a visualisation system concentrate on only the most important information and detect trends in it, which is precisely the idea behind data visualisation. Since humans have an exceptional ability to recognise patterns in images [16], visualisation is a good way to present large quantities of data and let human operators find potentially interesting trends in it.

Overall, visualisation is a dynamic process and the way users interact with the displayed data will vary for each individual, which is why it is important for a developer not to get attached to one mode of thinking or presenting data, as it is unlikely that all users of the system will also share this mentality. The way the users are going to interact with the system needs to be studied and the application needs to be designed accordingly [29].

4.3 Conclusion and Future Work

Here, the overall summary of the findings of this project are presented. In addition to that, several future research directions and improvements to the proposed system are suggested.

4.3.1 Conclusion

Overall, the issues in the field of network and security log data analysis and visualisation have been explored in this report. Particular attention has been paid to the design of a prototype 3D network traffic and security alert visualisation system which utilises a novel approach to the display of such data. The visualisation application proposed here has been described in detail and its limitations have been noted. Finally, a set of guidelines for the design of any such future system is suggested.

The research presented in this paper unifies several aspects of previous works and combines different strategies for dealing with large amounts of log data from network and security devices, their processing and presentation. It presents a framework for variable-precision grouping of network and security log data in memory and explores a novel three-dimensional network traffic and security alert visualisation technique.

4.3.2 Future Work

Firstly, the need for sufficiently advanced visualisation systems is growing, and therefore it would be practical to adapt the proposed system into a general-purpose visualisation application, with the purpose of extending the range of data that can be processed and displayed, from purely network and security information to other scientific data.

Secondly, the most useful future research direction is the integration of the proposed system with information from other sources such as other types of firewall and router devices. This will enable visual mapping of links and relationships (for example, between machines and network nodes) and improve the likelihood of security alerts and intrusions being detected. This ability to correlate and visually display information from multiple sources would be highly beneficial for many applications in the areas of computer security and network traffic analysis, and indeed such correlation ability is the basis of many new commercial SIEM (Security Information Event Management) systems.

Furthermore, addition of real-time data processing capabilities to the proposed system has been considered. While it has not been implemented, the ability to dynamically generate graphs based on real-time information coming into the system is highly valuable, turning a static log analysis application into a very capable and helpful security and network analysis tool.

Finally, as described in Section 4.1, a comprehensive user study is desirable and would be of great value to any continued research in the area of 3D network traffic and security alert visualisation. The most practical methods could be found and the less efficient and unintuitive graphing techniques would become known, so that their use can be avoided or limited. This report proposes theoretical ways of dealing with some common problems in the field of network and security log data visualisation, but the actual usefulness of such methods must be verified by further research.

Bibliography

[1] Robert Ball, Glenn A. Fink, and Chris North. Home-centric visualization of network traffic for security administration. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 55–64, New York, NY, USA, 2004. ACM. ISBN 1-58113-974-8. doi: http://doi.acm.org/10.1145/1029208.1029217.

[2] Kevin K. Chang, Nithya Ramanathan, Deborah Estrin, and Jens Palsberg. D.a.s.: deployment analysis system. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 301–301, New York, NY, USA, 2005. ACM. ISBN 1-59593-054-X. doi: http://doi.acm.org/10.1145/1098918.1098966.

[3] Kenneth C. Cox, Stephen G. Eick, and Taosong He. 3d geographic network displays. *SIGMOD Rec.*, 25(4):50–54, 1996. ISSN 0163-5808. doi: http://doi.acm.org/10.1145/245882.245901.

[4] S. Eick and A. Karr. Visual scalability. *Journal of Computational and Graphical Statistics*, 11 (1):22–43, 2002.

[5] Stephen G. Eick, Michael C. Nelson, and Jerry D. Schmidt. Graphical analysis of computer log files. *Commun. ACM*, 37(12):50–56, 1994. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/198366.198378.

[6] G. Conti et al. Countering security information overload through alert and packet visualization. *Computer Graphics and Applications, IEEE*, 26(2):60–70, 2006. doi: 10.1109/MCG.2006.30.

[7] P. C. Wong et al. Dynamic multiscale magnifying tool for exploring large sparse graphs. *Information Visualization*, 7(2), 2008.

[8] OpenGL: The Industry Standard for High Performance Graphics. Retrieved on 29 October 2009, from http://www.opengl.org/.

[9] R.M. Friedhoff and M.S. Peercy. *Visual Computing*. Scientific American Library, New York, 2000.

[10] Patrick Hertzog. Visualizations to improve reactivity towards security incidents inside corporate networks. In *VizSEC '06: Proceedings of the 3rd international workshop on Visualization for computer security*, pages 95–102, New York, NY, USA, 2006. ACM. ISBN 1-59593-549-5. doi: http://doi.acm.org/10.1145/1179576.1179596.

[11] T. J. Jankun-Kelly, Kwan Liu Ma, and Michael Gertz. A model for the visualization exploration process. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 323–330, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7803-7498-3.

[12] Eleftherios E. Koutsofios, Stephen C. North, Russell Truscott, and Daniel A. Keim. Visualizing large-scale telecommunication networks and services (case study). In *VIS '99: Proceedings of the conference on Visualization '99*, pages 457–461, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press. ISBN 0-7803-5897.

[13] Gregory Kramer. *Auditory Display: Sonification, Audification and Auditory Interfaces*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000. ISBN 0201626039.

[14] GLUI User Interface Library. Retrieved on October 28 2009, from: http://glui.sourceforge.net/.

[15] N. Lopez M. Kreuseler and H. Schumann. A scalable framework for information visualization. In *Proceedings of the IEEE Symposium on information Vizualization 2000*, Washington, DC, 27, USA, October 09-10 2000. IEEE Computer Society.

[16] Kwan-Liu Ma. Cyber security through visualization. In *APVis '06: Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation*, pages 3–7, Darlinghurst, Australia,

Australia, 2006. Australian Computer Society, Inc. ISBN 1-920682-41-4.

[17] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman and Co, San Francisco, 1982.

[18] Thomas L. Naps, Guido Rossling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, and J. Ángel Velazquez-Iturbide. Exploring the role of visualization and engagement in computer science education. In *ITiCSE-WGR '02: Working group reports from ITiCSE on Innovation and technology in computer science education*, pages 131–152, New York, NY, USA, 2002. ACM. doi: http://doi.acm.org/10.1145/960568.782998.

[19] Christos Papadopoulos, Chris Kyriakakis, Alexander Sawchuk, and Xinming He. Cyberseer: 3d audio-visual immersion for network security and management. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 90–98, New York, NY, USA, 2004. ACM. ISBN 1-58113-974-8. doi: http://doi.acm.org/10.1145/1029208.1029223.

[20] A. Perer and B. Shneiderman. Balancing systematic and flexible exploration of social networks. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):693–700, 2006.

[21] Pin Ren, John Kristoff, and Bruce Gooch. Visualizing dns traffic. In *VizSEC '06: Proceedings of the 3rd international workshop on Visualization for computer security*, pages 23–30, New York, NY, USA, 2006. ACM. ISBN 1-59593-549-5. doi: http://doi.acm.org/10.1145/1179576.1179582.

[22] Asaf Shabtai, Denis Klimov, Yuval Shahar, and Yuval Elovici. An intelligent, interactive tool for exploration and visualization of time-oriented security data. In *VizSEC '06: Proceedings of the 3rd international workshop on Visualization for computer security*, pages 15–22, New York, NY, USA, 2006. ACM. ISBN 1-59593-549-5. doi: http://doi.acm.org/10.1145/1179576.1179580.

[23] Ben Shneiderman. Extreme visualization: squeezing a billion records into a million pixels. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 3–12, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-102-6. doi: http://doi.acm.org/10.1145/1376616.1376618.

[24] T. Takada and H. Koike. Tudumi: information visualization system for monitoring and auditing computer logs. In *Information Visualisation, Proceedings, Sixth International Conference*, pages 570–576, 2002. doi: 10.1109/IV.2002.1028831.

[25] GLUT: The OpenGL Utility Toolkit. Retrieved 29 October 2009, from: http://www.opengl.org/resources/libraries/glut/.

[26] M. Eduard Tudoreanu. Designing effective program visualization tools for reducing user's cognitive effort. In *SoftVis '03: Proceedings of the 2003 ACM symposium on Software visualization*, pages 105–ff, New York, NY, USA, 2003. ACM. ISBN 1-58113-642-0. doi: http://doi.acm.org/10.1145/774833.774848.

[27] Craig Upson, Thomas A. Faulhaber Jr., David Kamins, David Schlegel David Laidlaw, Jeffrey Vroom, Robert Gurwitz, and Andries Van Dam. The application visualisation system: a computational environment for scientific visualisation. *IEEE Computer Graphics and Applications*, 9(4):30–42, July 1989.

[28] Jaime Urquiza-Fuentes and J. Ángel Velazquez-Iturbide. A survey of successful evaluations of program visualization and algorithm animation systems. *Trans. Comput. Educ.*, 9(2):1–21, 2009. doi: http://doi.acm.org/10.1145/1538234.1538236.

[29] Eliane R. A. Valiati, Carla M. D. S. Freitas, and Marcelo S. Pimenta. Using multidimensional in-depth long-term case studies for information visualization evaluation. In *BELIV '08: Proceedings of the 2008 conference on BEyond time and errors*, pages 1–7, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-016-6. doi: http://doi.acm.org/10.1145/1377966.1377978.

[30] D. Weiskopf. On the role of color in the perception of motion in animated visualizations. In *Proceedings of the Conference on Visualization '04*, pages 305–312, Washington, DC,

USA, October 10-15 2004. IEEE Computer Society. doi: http://dx.doi.org/10.1109/VIS.2004.73.

[31] J.M. Wolfe. *Visual search*. Addison-Wesley Longman Publishing Co., Inc., University College London Press, London, U.K., 1998.

A Appendix A - Screenshots

Figure A.1: Main program window with annotations and explanations of UI elements.

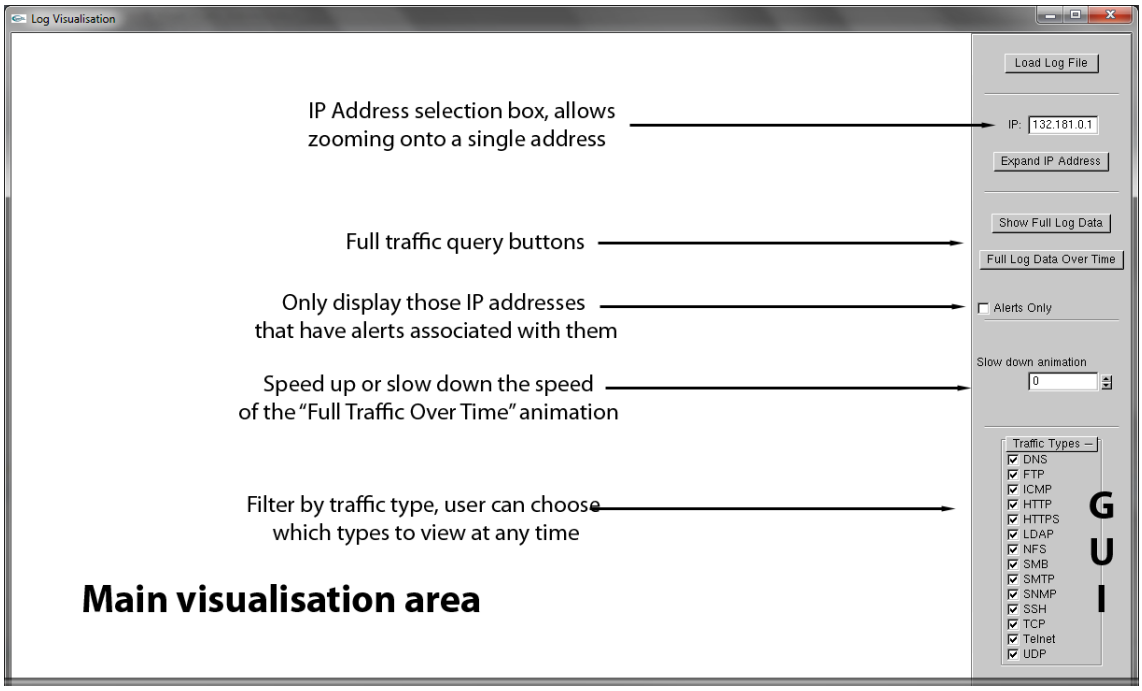


Figure A.2: Example of a full traffic query. IP addresses are located on the x-axis, the list of traffic types is placed on the z-axis and the amount of bytes is shown by the bar height. The back wall displays the total amount of traffic and incoming and outgoing traffic is colour coded. Light blue is being used to highlight the IP address currently selected. One alert is shown, caused by SMTP traffic for IP address 132.181.0.8.

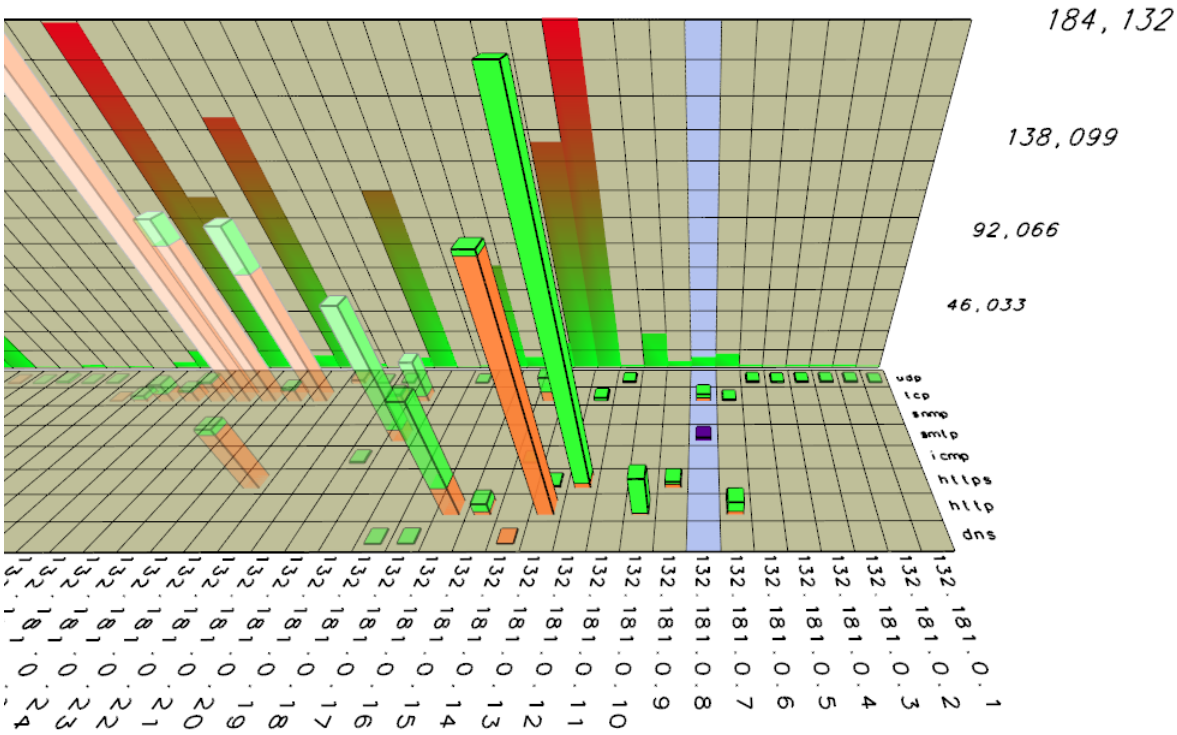


Figure A.3: Different view of the same data. The graph is dynamically split in the middle in order to draw the labels on the y-axis. Faraway bars are drawn with increased transarency. All bars are scaled according to the highest bar.

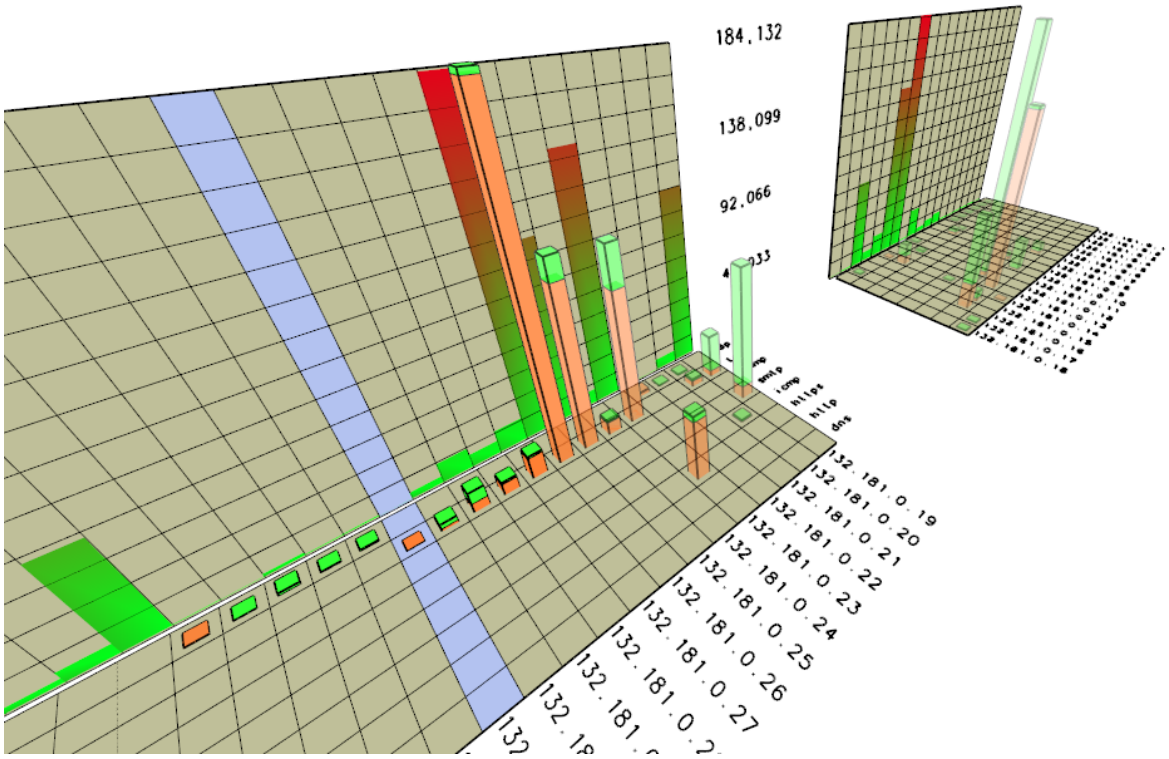


Figure A.4: Here a larger data set is shown. The traffic is separated by time intervals. Bars are being drawn on top of one another, with the lowest bars representing the oldest traffic and those above them showing more recent traffic. In the bottom left corner the current time interval is shown. All bars are drawn dynamically and animated over several seconds.

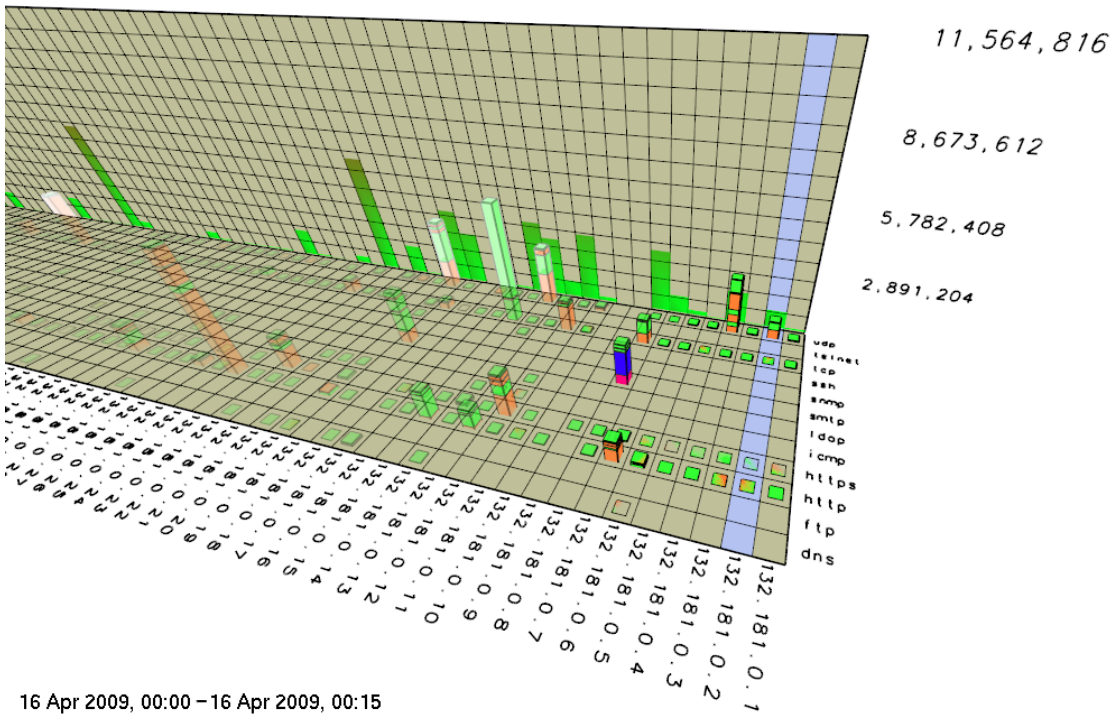


Figure A.5: Here all traffic for one local IP address is shown. The visualisation style is the same as when a full traffic query for the local network is performed. However, in this case, instead of local IP addresses being shown on the x-axis, all remote IP addresses that this particular local address had contact with are shown.